Mitigation and Improving SHA-1 Standard Using Collision Detection Approach

Zeyad Al-Odat
North Dakota State University
Fargo, ND, USA
zeyad.alodat@ndsu.edu

Mazhar Ali
COMSATS Institute of Information Technology
Abbottabad, Pakistan
mazhar@ciit.net.pk

Samee U. Khan
North Dakota State University
Fargo, ND, USA
samee.khan@ndsu.edu

***Abstract*—This paper introduces a collision detection methodology and an improved version of Secure Hash Algorithm (SHA-1) standard. The proposed work helps to protect weak primitives from any possible collision attack. Two designs are implemented to help protect and improve SHA-1 standard. The first design employs near collision detection approach that was proposed by Marc Stevens. The second design is the proposed work that employs two block calculation schemes. Both designs are tested and verified for examples of collided messages. The designs can detect the collision probability and produce a different hash for weak messages that are susceptible to collision attack.**

***Index Terms*—Hash; SHA-1; Collision; Attack.**

I. Introduction

Secure Hash Algorithms (SHA) was started in 1991 with the first MD4 hash function by Rivest [1]. An optimized version, MD5 was released two years later [2]. An improved version with longer hash length (SHA-0) was developed in 1993, and for security issues, SHA-1 was announced as the official standard in 1995 by National Institute for Science and Technology (NIST) [3].

Cryptographic hash functions are widely used in applications, starting from simple password implementation to message authentication over unsecure network. The cryptanalyst tries to verify the strength of secure hash algorithm functions by many tools and techniques. Three challenges exist to verify the completeness of any hash standard (preimage, 2nd preimage and collision resistances). Preimage resistance property means to easily obtain the hash from given message, but difficult to extract it back from a given hash. Second preimage resistance means that it is difficult to find two messages M1 and M2 generating the same digest (Hash). While collision resistance property means that the hash function resists the probability to generate the same output hash for two messages or more, even though they are different or equal [4].

SHA-1 is still used by many entities for data authentication schemes, such as digital signature authentication. However, many researches proved that SHA-1 is exposed to collision attack through a thoroughly efforts to find any hiatus lead to break the hash system [5]–[7]. For instance, the collision attack for SHA-1 hash standard has gradually been obtained, such as 40-steps collision [8], 53-steps [9], 64-steps by Canniere *et. al.* [10], 76-steps [7], and 80-steps full collision attacks as presented by Wang *et. al.* in [11].

However, in the situation of weak messages the old hash case appears. As some old hashes are not replaceable to a new secure standards. Moreover, the data verifiers continue to accept weak and malicious messages for long time to come [12]. Therefore, the need for an improved version of SHA-1, or a counter collision method arose to protect primitives that are still using SHA-1 standard.

In this paper, we are proposing a counter cryptanalyst technique to help protect SHA-1 standard against collision attack. The rest of paper is organized as follow. Section 2 gives a brief description about SHA-1 and collision attack. In Section 3 literature review of previous works is discussed. The proposed algorithm and methodology are presented in Section 4. Section 5 concludes the paper.

II. Preliminaries

Some preliminaries need to be addressed to better understand the proposed system. In the subsequent text a brief description of SHA-1 standard, SHA-1 collision attack, and threat model are presented.

A. Brief Description of SHA-1

SHA-1 standard follows Merkle-Damgard (MD) structure [13]. SHA-1 function takes the message of length less than 2^{64}, and divide it into equal blocks, then process them sequentially. SHA-1 gives 160 bits output digest, to do so the message goes through several steps and compression operations before the output hash is produced, as can be seen in Figure 1.

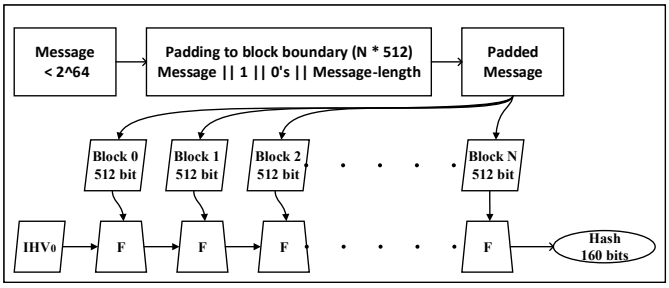

Fig. 1. SHA-1 General structure, takes message of length $< 2^{64}$, padd it, then divide it into equal size blocks.

The whole process of SHA-1 compression function is summarized by the following steps:

1) **Message padding**: In this step a '1' is appended at the end of the input message M and followed by least number of 0's until it congruent to 448 Mod 512. The size of the original message M is appended as big-endian 64-bits, as seen in Figure 2. Then the resulting padded message becomes $\widehat{M} = N*512$, for real $N \geqslant 1$.

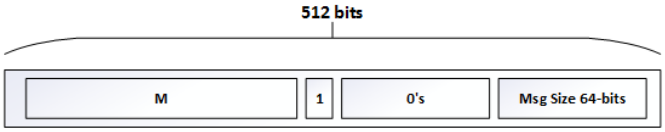

Fig. 2. Message padding mechanism

2) **Message divide**: Divide the padded message $\widehat{M}$ into N 512-bit blocks $M_0, M_1, ..., M_{N-1}$.
3) **Initial Hash Values**: SHA-1 needs five 32 bits IHVs (H0, H1, H2, H3 and H4) initialized with fixed values (67452301_{16}, $EFCDAB89_{16}$, $98BADCFE_{16}$, 10325476_{16}, $C3D2E1F0_{16}$) respectively. Moreover five 32 bits working state variables (A,B,C,D and E) initialized with the values of IHVs accordingly.
4) **Processing**: To compute the hash value of N blocks message, the process goes through SHA-1 compression function for N+1 states (state for each block plus the initial state), starting with IHV.
5) **Message Expansion and Working State Variables**: The divided block of sixteen 32 bit words are used to expand the message block from 512 to 2560 bits using message expansion Equation 1. The 16 message block chunks are initialized with the messages block, then they are used to calculate the rest of expansion equation to get chunks W[17] to W[79].

$$W_j = \begin{cases} M_i^{(j)} & ,0 \leq j \leq 15 \\ (W_{j-16} \oplus W_{j-16} \oplus W_{j-16} \oplus W_{j-16} \oplus)_{\lll 1} & ,16 \leq j \leq 79 \end{cases}. \quad (1)$$

Where, M_i^j is the j^{th} chunk of block i, $\oplus$ logical XOR, and $\lll_n$ left rotation by n-bit.
6) **Internal Process**: The compression function consists of 80 steps. Each 20 steps conform a round, and each round has distinct round function and constant K_i as seen in Table I. The functions are used to calculate the step output. The working state variables change after each step of the 80-steps according to the values below.

$$\begin{aligned} A_t &= RL^5(A_{t-1}) + F_t(B_{t-1}, C_{t-1}, D_{t-1}) + E_{t-1} + W_t + K_t \\ B_t &= A_{t-1} \\ C_t &= RL^{30}(B_{t-1}) \\ D_t &= C_{t-1} \\ E_t &= D_{t-1}. \end{aligned}$$

Where, $RL^n(x)$ is a left rotation of word x by n-bits, and $F(B, C, D)$ is the logical function equation.
7) **Hash Output**: The output hash is calculated by adding the output of the last step to the initial hash values (H0, H1, H2, H3 and H4). If the current block is the last block, then the output hash is the concatenation of the five hash values together. Otherwise the new IHV will be fed as an initial value for the next block calculations. The output hash is represented as:

$$H0 \parallel H1 \parallel H2 \parallel H3 \parallel H4.$$

TABLE I
SHA-1 ROUND FUNCTIONS AND CONSTANTS

Round and Steps	Round Function F (B,C,D)	Round Constant (Ki)
Round1 (0-19)	$(B \wedge C) \vee (\neg B \wedge D)$	*0x5A827999*
Round2 (20-39)	$(B \bigoplus C \bigoplus D)$	*0x6ED9EBA1*
Round3 (40-59)	$(B \wedge C) \vee (B \wedge D) \vee (C \wedge D)$	*0x8F1BBCDC*
Round4 (60-79)	$(B \bigoplus C \bigoplus D)$	*0xCA62C1D6*

B. SHA-1 Differential Attack

The main goal of SHA-1 collision attack is to find two or more messages that lead to the same output hash. All current researches on finding the collision are basically built on the methodology described by Wang *et. al.* [11]. Wang's finding drew the beginning road to the real collision project that is recently announced by the collaboration of Marc *et. al.* and Google [6]. The aforesaid project found the first real collision attack for the SHA-1 hash function [6], [7], [14].

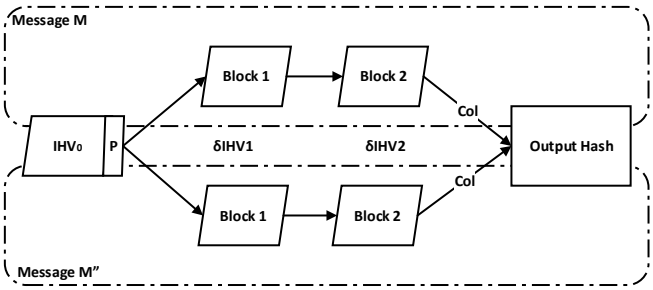

Fig. 3. Two blocks collision attack. Two messages start with same prefix (IHV), after two blocks processing the result end with a collision.

The procedure depends on finding a disturbance vector which includes a real path among compression operations that lead to collision. Disturbance vector is an (80×32) bit vector that contains the modular differences between Message M and $\widehat{M}$ (so that it called differential attack). Figure 3 gives an explanation of two blocks collision attack. For both messages M and $\widehat{M}$ the differences are calculated after each step. These differences are constructed from the internal block differences after each step of the compression calculation steps. Each difference must meet predefined conditions, once met the block difference is added to the disturbance vector. For more details about disturbance vector and differential attack the reader advised to read these resources [5], [11], [15].

C. Threat Model

The threat model is the general data flow from attacker perspective. Figure 4 shows a general thread model for a document that contains essential information. The Human user

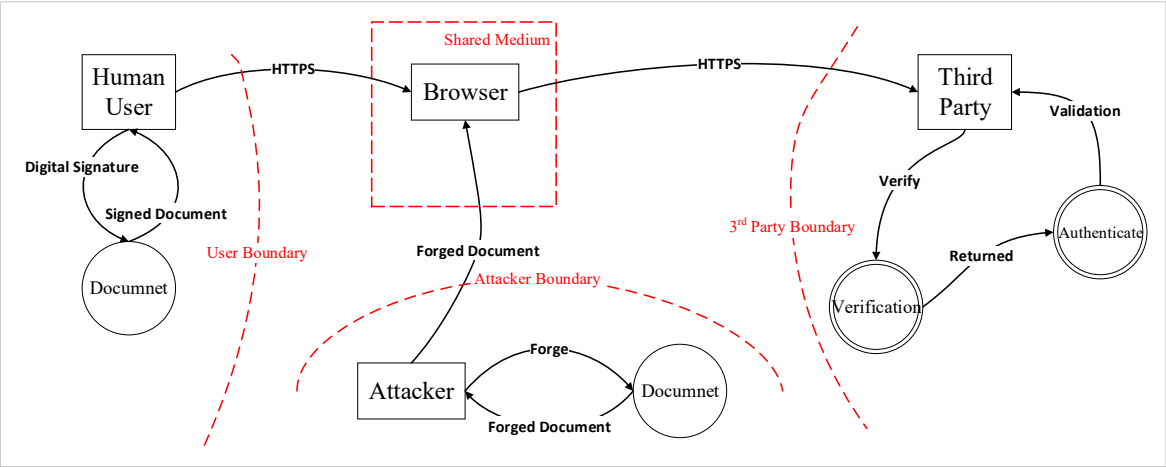

Fig. 4. Threat model of the proposed system

through operating system process digitally sign a document. Moreover, this signature is the SHA-1 hash of the document. The document is sent through Internet to a third party. The attacker in the middle crafted a message to produce the same hash value with different information than the original owner. The third-party user receive the document for verification. The document from the attacker is verified and authorized as it contains a valid hash value.

III. Literature Review

The first differential attack was proposed by Chabaud and Joux [16]. They were the first to introduce the idea of differential path taking the XOR differences between messages, as seen in Figure 5. Their work was applied on two message's blocks, and the XOR difference is calculated after each round of computations. The messages were prone to collision by reducing the XOR difference, $\Delta 2 = 0$. Another convenient approach was the work presented by Wang *et. al.* [17]. Wang presented an attack to different hash functions (MD4, MD5, HAVAL, RIPEMD and SHA-0), they were able to break all of them. The attack was basically dependent on modular difference (not the XOR differences previously proposed in [16]). Also using message modification techniques these differences were made equal to zero. The result was a feasible attack on all aforesaid hash functions. On the other hand, in [5] Wang *et. al.* proposed an improved version of the previously proposed attack on SHA-0 by building the differential path according to pre-specified conditions.

Manuel *et. al.* in [18] proposed SHA-0 attack within one hour. The work relied on the same concept of differential attack from [17]. The author got benefited from the Boomerang Attack by finding optimal differential vector rather than the one used by Wang *et. al.* The proposed work found an attack with a claimed complexity of $2^{33.5}$, and one hour of operation to get the final hash neglecting the time used to compute the optimal differential vector. Following the same approach Wang *et. al.* in [11] announced a new collision attack on full SHA-1 with complexity 2^{69}. Their approach mainly depends on the

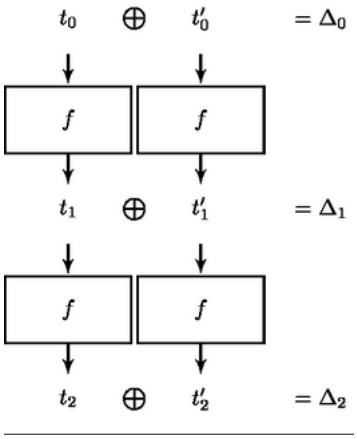

Fig. 5. The goal here is to find a difference in the input $\Delta 0$ such that after some iterations getting $\Delta 2 = 0$, in other words no difference [16].

previous research on SHA-0 and MD5, using differential path of modular differences to construct the disturbance vector.

The big contribution on this area was the work done by Marc *et. al.* [19]. Their work lead to the real collision attack for SHA-1. The work was the base for the collision attack for SHA-1 that has been lately published [6]. The attack generates the same hash from two different files.

The efforts to counter a collision attack was discussed in few literature [12], [20], [21]. In [20] a method for detecting and preventing collision of hashes during data transmission was presented. The authors registered their findings as a patent with number: $US8,086,860-B2$. The proposed work comprises of four steps: shuffling of bits, compression, T-functioning, and linear feedback shift registration (LFSR). The framework adds more haphazardness to the produced hash information to avoid collision. Notwithstanding, the produced hashes differ from if they were generated from the original SHA-1, regardless of whether the original hash out of collision suspicious. Stevens *et. al.* in [12] proposed an algorithm to detect the occurrence

of collision for the Flame attack. Flame attack is a kind of collision that is used to breach windows update patch. Their work depends on the previous published disturbance vectors that leads to a collision attack. The authors used the top published disturbance vectors to detect the collision before it takes place and invalidates the output hash once the message is marked as a suspicious one. The proposed work is compatible with MD5 and SHA-1 hash standards. lately, Stevens *et. al.* in [21] proposed a speedup mechanism to detect SHA-1 collision attack based on unavoidable bit condition. Their work lead to a significant speedup over the previous proposed work [12]. The claimed speed is 1.96 times slower than the original SHA-1 compression function.

The main goal of our work is to protect the entities that still use SHA-1 hash function in their architecture. For backward compatibility there is a need to detect collisions for SHA-1.

IV. Proposed Methodology

The idea of counter SHA-1 collision attack is depicted in Figure 6. The input message is processed and checked using collision detection mechanism, that we will discuss later. Once the algorithm returns true, then the output hash will be the truncated SHA-512/160, otherwise the regular SHA-1 hash will be passed to the output.

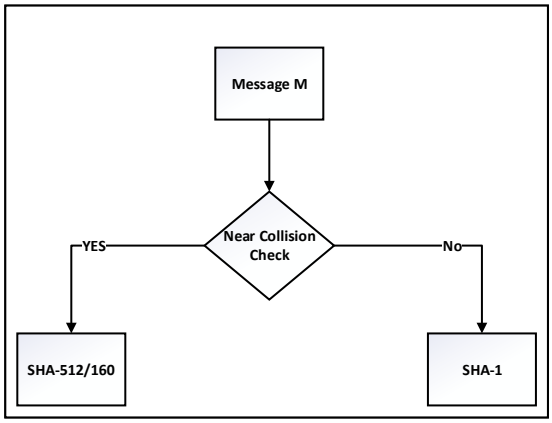

Fig. 6. General architecture for the improved SHA-1

Collision detection algorithm returns True if a collision attack is detected, and false otherwise. Once a message that crafted using a collision attack is detected, then the algorithm will decline the hash output from the regular SHA-1. In our design we chosen SHA2-512 to replace the hash calculation of weak messages because SHA2-512 is the strongest version of SHA-2 standard. Moreover, SHA-2 standard follows the same construction model as SHA-1 [22].

A. Proposed Work

We are presenting two approaches for SHA-1 counter collision attack. The first one relying on Marc Stevens approach presented in [12]. The second design is our proposal. But before we go through the details of both approaches we need to give insights about backward expansion equation.

1) Backward Expansion Equation: The backward expansion equation was defined by Manuel *et. al.* in [23]. According to Mauel *et. al.* the expansion of message can be calculated by either side, and the initial blocks (IHV) can be extracted if sufficient working state variables are available. Equation 2, shows the backward expansion equation, as can be noticed it uses the same number of terms as forward expansion Equation 1. Backward expansion equation differs from the standard way of SHA-1 expansion as it expands backward to obtain ($W_{-64,...,W_{-1}}$).

$$W_{i-16} = RR(W_t, 1) \oplus W_{t-3} \oplus W_{t-8} \oplus W_{t-14}. \quad (2)$$

Where, t values between $16 \leqslant t \leqslant 79$.

When both expansion equations (backward and forward) combined together the list of 144-32-bits words are constructed as follow:

$$W_{-64}, ..., W_{-1}, W_0, ..., W_{15}, W_{16}, ..., W_{79}.$$

According to the aforesaid combination, any sequence of eighty consecutive 32-bit words is a valid expansion [23].

2) First Approach (Employing Marc Stevens Mechanism): In this design the approach presented by Marc *et. al.* in [12] is employed to detect the collision attack. Figure 7 shows the working of approach. The process starts with any IHV_k (no need to be the first one). IHV_k is passed to the working state variable[1]. After processing working state variables the IHV_{k+1} is obtained. Processing steps include: message expansion, 80-steps compression function calculation and message differences comparison. After each step of the 80 steps the value of WS_i is used alongside with the values of tuple $((\delta B, i, \delta WS_i))$ to generate the sibling message $\widehat{M}$. Where δB is the block difference, δWS_i working state difference , i is the current step. For each possible combination of triples $(\delta B, i, \delta WS_i)$, the sibling message $\widehat{M_i}$ is extracted. From the extracted sibling message $\widehat{M_i}$, the compression function calculations are carried out to compute $IHV^{'}_{k+1}$. Then, both values $IHV^{'}_{k+1}$ and IHV_{k+1} are compared together for equality. Once the two values of IHV_{k+1} and $IHV^{'}_{k+1}$ are equal, then the original message is crafted with collision attack and the output hash will be declined. After that for the messages that give a match possibility for their hashes, the SHA-512 truncated to 160 is computed for both messages to get different hash values.

3) Second Approach: According to Wang *et. al.* all disturbance vectors were built on the two blocks near collision [11]. We can get benefited from this idea and instead of comparing the block differences after each step in the compression calculation (as stated by Marc Stevens), we propose the idea in Figure 8. The methodology goes through

[1]Marc *et. al.* called them WS_s, and we pointed to them with letters A, B, C, D, and E

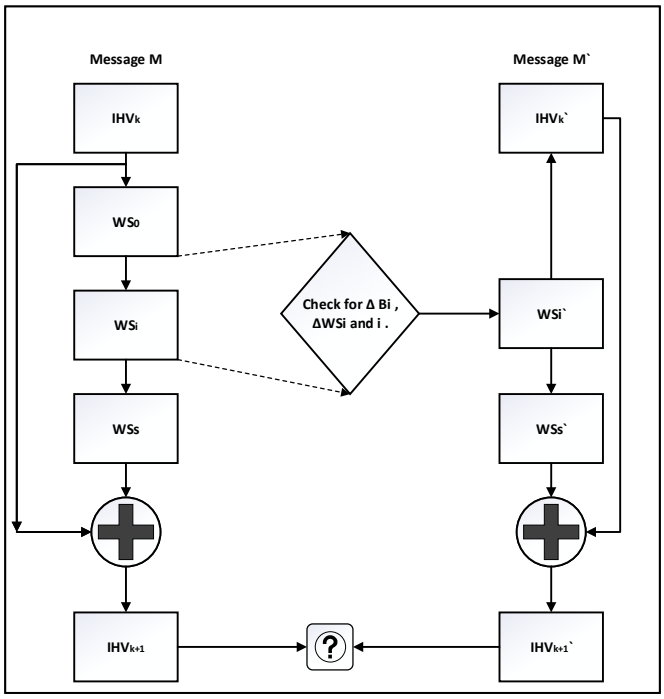

Fig. 7. First approach: Marc Stevens collision detection mechanism.

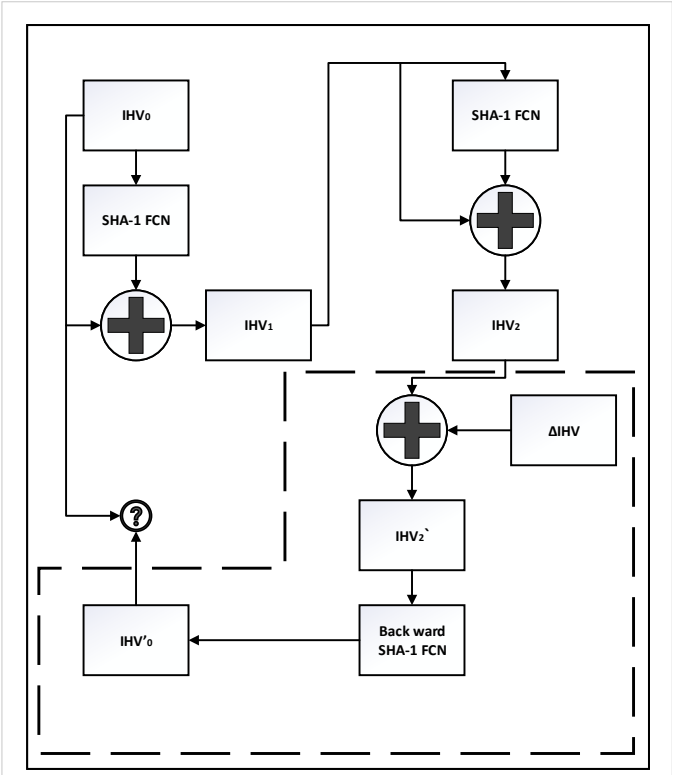

Fig. 8. Second approach: Proposed Collision Detection mechanism.

the compression calculations for two blocks. The process starts with IHV_0 to calculates IHV_1 through 80-steps calculation compression function. IHV_1 used as an input to the second block calculation to get IHV_2. For messages that were crafted with a collision attack the backward computation of $IHV_2^{'} = \delta IHV + IHV_2$ will gives us $IHV_0^{'}$ equal to IHV_0. Therefore, for all previously published disturbances the output values of the second block are added to the values of each disturbance vectors δIHV one at a time (The Dotted area in Figure 8). Then calculate the backward expansion equation as depicted in Equation 3. Applying backward expansion equation gives us the new value $IHV_0{'}$ that is compared with the original IHV_0. If both values of IHV_0 and $IHV_0{'}$ are equal then a message with collision is detected. After a collision is detected the truncated SHA-512/160 is calculated to replace the hash value of weak message.

$$W_{i-16} = RR(W_t, 1) \oplus W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \qquad (3)$$

In the second approach, instead of reproducing the message $M^{'}$ we only generate the initial hash value of the sibling message without any need to have the message itself or having IHV_{k+1}. This saves time and can be used to search for message siblings that might collide with the original one.

V. Results and Discussions

Both of the above-mentioned approaches were tested and verified for part of published disturbance vectors. Regarding the disturbance vectors that previously published by Wang *et. al.* [11], both algorithms work and give the hash value SHA-512/160 for the selected disturbance vector.

TABLE II
EXAMPLE OF TWO MESSAGES THAT COLLIDE AT STEP 58 [11] (ALL VALUES ARE IN HEXADECIMAL FORMAT)

M :	132b5ab6 a115775f 5bfddd6b 4dc470eb 0637938a 6cceb733 0c86a386 68080139 534047a4 a42fc29a 06085121 a3131f73 ad5da5cf 13375402 40bdc7c2 d5a839e2
M':	332b5ab6 c115776d 3bfddd28 6dc470ab e63793c8 0cceb731 8c86a387 68080119 534047a7 e42fc2c8 46085161 43131f21 0d5da5cf 93375442 60bdc7c3 f5a83982
Hash :	9768e739 b662af82 a0137d3e 918747cf c8ceb7d4

Table II represents two messages that collide to the same output hash after 58-steps. Our proposed designs can be applied for the shown messages. After applying the collision detection approaches we were able to detect the collision possibility between the two messages. In spite of the two input messages look close to each other with minor changes. Tables III and IV represent the hash values for the two PDF

TABLE III
SHA-1 VALUES OF TWO PDF FILES REPORTED BY MARC. *et. al.*

shattered-1.pdf :	38762cf7 f55934b3 4d179ae6 a4c80cad ccbb7f0a
shattered-2.pdf :	38762cf7 f55934b3 4d179ae6 a4c80cad ccbb7f0a

files that reported by Marc *et. al.* as an example of real SHA-1 collision. The two files are available for open access in [24], and the result of computing SHA-1 hash for both of them lead to the same output hash. After applying the collision detection

mechanisms for the two files, a collision warning is generated for the regular hash calculation. Then, the truncated SHA-512 to 160 hash value will be produced instead SHA-1.

TABLE IV
SHA-1 (SHA-512/160) USING PROPOSED ALGORITHMS

shattered-1.pdf :	42180282 5d3587fe 185abf70 9669bb96 93f6b416
shattered-2.pdf :	1af8b65d 6a0c1032 e00e48ac e0b4705e edcc1bab

For the proposed schemes it is possible to add more disturbance vectors to the detection algorithms whenever the new published vectors lead to a collision. Moreover, each disturbance vector has a probability of order 2^{-70} false positive occurrence [12], where a message is considered weak while it is not. But, with such probability the false positive occurrence is negligible. The designs were tested on laptop computer with Intel(R) Core (TM) i7-2640M CPU @ 2.80GHz and 8GB of RAM Under Linux (Ubuntu 18.04 LTS) platform. The testing codes were written under C programming language. To validate our scheme an example with a real collided messages that were built with collision attacks are fed to the designs to test the validity of the system. We were able to detect the collision and produce a safe hash values for the weak messages as depicted in Tables II, III, and IV.

VI. CONCLUSIONS

We presented two methods to improve SHA-1 standard against collision attack. The first design relies on Stevens's approach for detecting SHA-1 collision attack, in which the input message is checked against collision possibility according to three values $(\delta W_s, \delta B_i, i)$. These values belong to the previously published works of disturbance vectors that leads to collision. After each round the system is checked for the aforesaid three values are used to extract the sibling message from the given one. Finally, compare $IHV_{k+1}, IHV_{k+1}^{'}$ of both messages if they were equal then the original message is crafted with collision forgery. The second approach is based on two blocks collision and the backward expansion calculation equation. The initial hash value (IHV_0) is processed using 80 steps SHA-1 function. Then applying backward expansion equation to get $IHV_0^{'}$. For the messages that crafted with collision attack, both IHV_0 and $IHV_0^{'}$ will be equal. The truncated SHA-512/160 is suggested to replace suspicious message's hash outputs.

ACKNOWLEDGMENTS

The work of Samee U. Khan was supported by (while serving at) the National Science Foundation. Any opinion, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] Ronald L. Rivest. The md4 message digest algorithm. In *Proceedings of the 10th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '90, pages 303–311, Berlin, Heidelberg, 1991. Springer-Verlag.

[2] Ronald Rivest. The md5 message-digest algorithm. Technical report, 1992.

[3] Patrick Gallagher and Acting Director. Secure hash standard (shs). *FIPS PUB*, pages 180–3, 1995.

[4] Muhammad Barham, Orr Dunkelman, Stefan Lucks, and Marc Stevens. New second preimage attacks on dithered hash functions with low memory complexity. In *International Conference on Selected Areas in Cryptography*, pages 247–263. Springer, 2016.

[5] Xiaoyun Wang, Hongbo Yu, and Yiqun Lisa Yin. Efficient collision search attacks on sha-0. In *Annual International Cryptology Conference*, pages 1–16. Springer, 2005.

[6] Marc Stevens, Elie Bursztein, Pierre Karpman, Ange Albertini, and Yarik Markov. The first collision for full sha-1. In *Annual International Cryptology Conference*, pages 570–596. Springer, 2017.

[7] Pierre Karpman, Thomas Peyrin, and Marc Stevens. Practical free-start collision attacks on 76-step sha-1. In *Annual Cryptology Conference*, pages 623–642. Springer, 2015.

[8] Eli Biham, Rafi Chen, Antoine Joux, Patrick Carribault, Christophe Lemuet, and William Jalby. Collisions of sha-0 and reduced sha-1. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 36–57. Springer, 2005.

[9] Vincent Rijmen and Elisabeth Oswald. Update on sha-1. In *Cryptographers' Track at the RSA Conference*, pages 58–71. Springer, 2005.

[10] Christophe De Canniere and Christian Rechberger. Finding sha-1 characteristics: General results and applications. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 1–20. Springer, 2006.

[11] Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding collisions in the full sha-1. In *Crypto*, volume 3621, pages 17–36. Springer, 2005.

[12] Marc Stevens. Counter-cryptanalysis. In *Advances in Cryptology–CRYPTO 2013*, pages 129–146. Springer, 2013.

[13] Jean-Sébastien Coron, Yevgeniy Dodis, Cécile Malinaud, and Prashant Puniya. Merkle-damgård revisited: How to construct a hash function. In *Annual International Cryptology Conference*, pages 430–448. Springer, 2005.

[14] Marc Stevens, Pierre Karpman, and Thomas Peyrin. Freestart collision for full sha-1. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 459–483. Springer, 2016.

[15] Nick Merrill. Better not to know?: The sha1 collision & the limits of polemic computation. In *Proceedings of the 2017 Workshop on Computing Within Limits*, pages 37–42. ACM, 2017.

[16] Florent Chabaud and Antoine Joux. Differential collisions in sha-0. In *Advances in Cryptology—CRYPTO'98*, pages 56–71. Springer, 1998.

[17] Xiaoyun Wang and Hongbo Yu. How to break md5 and other hash functions. In *Eurocrypt*, volume 3494, pages 19–35. Springer, 2005.

[18] Stéphane Manuel and Thomas Peyrin. Collisions on sha-0 in one hour. *Lecture Notes in Computer Science*, 5086:16–35, 2008.

[19] Marc Stevens. New collision attacks on sha-1 based on optimal joint local-collision analysis. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 245–261. Springer, 2013.

[20] Natarajan Vijayrangan. Method for preventing and detecting hash collisions of data during the data transmission, December 27 2011. US Patent 8,086,860.

[21] Marc Stevens and Daniel Shumow. Speeding up detection of sha-1 collision attacks using unavoidable attack conditions. *IACR Cryptology ePrint Archive*, 2017:173, 2017.

[22] Quynh H Dang. Secure hash standard. Technical report, 2015.

[23] Stéphane Manuel. Classification and generation of disturbance vectors for collision attacks against sha-1. *Designs, Codes and Cryptography*, 59(1-3):247–263, 2011.

[24] www.shattered.io, Jul 2017. [Online; accessed 8. Jul. 2018].